Apple II
Technical Notes

Developer Technical Support

Apple IIGS
#60: Menu Manager Memorabilia

Revised by: Matt Deatherage, Dave Lyons, & Tim Swihart November 1990
Written by: Dave Lyons July 1989

This Technical Note discusses the Menu Manager, specifically a few anomalies and some tips for making menus your friends.
Changes since May 1990: Noted that System Software 5.0.3 fixes a bug in `NewMenuBar2`.

The Menu Manager Is Your Friend

In general, this is the truth. You can do all kinds of nifty things with menus, especially in System Software 5.0 and later. However, there are a few things you should know unless you generally are fond of pain in your life.

Disabling Menus Gracefully

As documented, `SetMenuFlag` can be used to disable and enable entire menus. When a menu is disabled, the menu title and all items within the menu are disabled. You may pull down a disabled menu, but you may not select any item within it (unless the routine `MenuGlobal` has been used to allow inactive menu items to be selected).

Volume 1 of the *Apple IIGS Toolbox Reference* says you should call `DrawMenuBar` if you change the appearance of a menu title with `SetMenuFlag`. You can do this; this is fine. It may, however, induce dizziness if used often.

A more graceful way to dim menus is to follow `SetMenuFlag` with `HiliteMenu`. Calling `HiliteMenu` causes the menu title to be redrawn to reflect the current (or new) highlighting and menu flags. Using `HiliteMenu` instead of `DrawMenuBar` allows you to disable and enable menus gracefully, without noticeable flicker or threat of nasty patent infringement lawsuits from strobe light manufacturers.

“System” Bars Versus “Window” Bars

As far as the Menu Manager is concerned, there are only two kinds of menu bars. One kind is in a window and the other kind is not. The former are called “window” menu bars and the latter are generally referred to as “system” menu bars.

Most people think of the System bar as the big menu bar across the top of the screen. This is encouraged by calls like `SetSysBar`, which takes a menu bar handle and sets the menu bar across the top of the screen to that menu bar. Trying to rename one or the other of these two concepts at this point is probably useless; instead, this Note refers to the bar across the top of the screen as the “System” bar (with a capital S), and menu bars not in windows as “system” bars (with a lowercase s).

When you start the Menu Manager, it creates an empty System bar for you. Before System Software 5.0, most people simply called `NewMenu` and `InsertMenu` to insert menus into that System bar. All was well in the world.

When 5.0 was released, it became very easy to create a new menu bar and all the menus within it using the `NewMenuBar2` call. This avoids a lot of code, and many new people use it. The problem comes with `DrawMenuBar`. If you simply call `NewMenuBar2` to obtain your menu bar and menus from resources, then call `DrawMenuBar` to make them visible, you usually get an empty menu bar. Why? The `windowPtr` parameter passed to `NewMenuBar2` determines whether or not the new menu bar created is a system bar or a window bar—it does **not** force the new bar to be the System (note the capital ‘S’) bar. So when `DrawMenuBar` draws the current System bar, it hasn’t changed from the empty default one created by `MenuStartUp`.

This is why Volume 3 of *Apple IIGS Toolbox Reference* recommends code similar to the following:

```
menuHandle := NewMenuBar2(refDesc,menuBarTRef,NIL);
SetSysBar(menuHandle);
SetMenuBar(NIL);        {NIL makes the System bar the current menu bar}
```

if you want your menu bar to be the one across the top of the screen.

A Bug in NewMenuBar2

`NewMenuBar2` is a handy thing to have around, but it does have a problem in 5.0.2 and earlier. When the Menu Manager is done with resources, it tries to use the internal toolbox call `CMReleaseResource` to free them in memory. However, it passes the wrong resource ID, and `CMReleaseResource` calls `SysFailMgr` if it encounters any errors at all (such as `Specified resource not found`).

What `NewMenuBar2` does improperly is push the high word of the resource ID onto the stack twice, instead of the high word followed by the low word. Because of the way the Resource Manager operates, `CMReleaseResource` returns with no error if the ID passed is `NIL`, but the resource is not released (another good reason not to try to use the illegal value `NIL` as a resource ID).

If the high word of the menu bar resource is $0000, `NewMenuBar2` passes a resource ID of `NIL` to `CMReleaseResource`, which then doesn't quite release the resource, but returns no error. The menu bar resource hangs around in memory until `ResourceShutDown`. It's usually fairly small, so this is no loss. It still takes up less room than menu strings, which had to stay in memory until `MenuShutDown`.

If the high word of the menu bar resource is **not** zero, the bug causes `CMReleaseResource` to bring down the system. When using System Software 5.0.2 or earlier, make sure all menu bar resource IDs have a high word of $0000. System Software 5.0.3 fixes this bug.

Menu and Menu Title ID Numbers

Table 13-4 in Volume 1 of *Apple IIGS Toolbox Reference* gives a listing of menu and menu item ID numbers. In both lists, $0000 and $FFFF are "reserved for internal use" and noted that $0000 usually indicates the first menu in the bar (or first item in the menu) and $FFFF usually indicates the last menu in the bar (or last item in the menu). Some developers have taken this to mean that they should give their first menu an ID of $0000 and their last one an ID of $FFFF.

This assumption is incorrect.. The Menu Manager may change these values internally to reflect such IDs, but they must not be assigned that way by an application. Some applications that use IDs of $0000 or $FFFF break under System Software 5.0 and later. Note that $0000 **can** be used as the `insertAfter` parameter to `InsertMenu` to insert a menu at the left of a menu bar, but $FFFF is not a valid `insertAfter` value.

Desk Accessories and Menus

Some desk accessory developers would like to have their NDAs insert a menu in the System menu bar. While the menu itself can be inserted, the NDA **cannot** detect that a user has selected an item within that menu. The application gets the event and does not know what to do with it. NDAs that need a menu can put a menu bar in their own window. Since the `mouseDown` event then happens within the NDA's window, the NDA gets the event and can handle it normally. Be sure to make the NDA's menu bar the current menu bar before calling `MenuSelect` from within your NDA (to avoid possible conflicts between NDA menu item IDs and application menu item IDs). Restore the current menu bar to the application's menu bar before returning control to the application. Failure to do so prevents the application from finding its menus. Apple IIGS Technical Note #3, Window Information Bar Use documents how to put a menu in a window's information bar.

Documentation Error in MenuSelect

Volume 1 of *Apple IIGS Toolbox Reference* states that `MenuSelect` returns the menu ID and the item ID of the selected item in the `when` field of the event record. This is incorrect. `MenuSelect` actually returns the information in the `wmTaskData` field of the **task record** (and this, in fact, is why you pass a task record and not just an event record to `MenuSelect`).

Menu Strings and Bank Boundaries

`NewMenu` takes a pointer to a string; this string must **not** cross a bank boundary. If it does, a menu containing random garbage may result.

If your `NewMenu` strings are contained in your code segments, everything is fine—code segments cannot cross bank boundaries. Depending on your development environment, strings that are not in a code segment may or may not be allowed to cross bank boundaries. If you can find no other way to guarantee the strings do not cross a bank boundary, use `NewHandle` to allocate blocks with attributes $4010 (fixed, no bank cross) and copy the strings to these blocks.

If you create menus from resources, be sure the resources have their `noCrossBank` attribute bits set. Note that a memory block that **can** cross a bank boundary usually does **not**, so your application may be working by accident.

Note that this restriction applies only to menu strings, not the menu templates that can be used with `NewMenu2`.

Return Values From GetMenuTitle and GetMItem

Starting with System Software 5.0, `GetMenuTitle` and `GetMItem` can return handles and resource IDs, not just pointers. The type of data returned depends on how the menu or item was created, so existing applications are not affected. For more information, see *Apple IIGS Toolbox Reference*, Volume 3, Chapter 37, "New Features of the Menu Manager."

Further Reference

- *Apple IIGS Toolbox Reference,* Volumes 1 & 3
- Apple IIGS Technical Note #3, Window Information Bar Use